RED ANNOTATED SPECIFICATION
OF
U.S. PATENT APPLICATION 09/301,438,
ATTORNEY DOCKET NO. NS-3799 US

DVD/DVB DECODER

Christopher K. Wolf

Ygal Arbel

Himanshu A. Sanghavi

FIELD OF THE INVENTION

The present invention relates to a decoder system and, more particularly, to a decoder system for decoding Digital Versatile Disc (DVD) or Digital Video Broadcast (DVB) data streams.

BACKGROUND

Digital Versatile Disc (DVD) and Digital Video Broadcast (DVB) data decoders are well known. A conventional DVD/DVB decoder system includes a central processing unit, a stream demultiplexer, an audio decoder, a video decoder, a memory controller, an MPEG system timer, a video output processor and an audio output processor.

Conventional DVD/DVB decoder systems have many disadvantages. First, they typically include two separate depacketizers; an audio depacketizer coupled to the audio decoder for depacketizing audio data, and a video depacketizer coupled to the video decoder for depacketizing video data, leading to duplication of many tasks. Therefore, such systems are not cost effective.

Second, the Central Processing Unit (CPU) of conventional DVD/DVB systems is frequently interrupted,

requiring the CPU to temporarily abandon its existing tasks to service the interrupts, leading to inefficient utilization of the CPU. Furthermore, because the interrupts are frequent and not synchronized to a

5 common event, such systems must meet strict and inflexible timing requirements. Moreover, in such systems, because the CPU typically demultiplexes the stream data in addition to handling many other tasks, it must be very powerful and hence relatively

10 expensive.

Third, the Packetized Elementary Stream (PES) data supplied to the audio and the video decoders of conventional DVD/DVB systems typically includes a time stamp which must be stripped off before the data is

15 decoded, resulting in processing inefficiencies.

SUMMARY

The Digital Versatile Disc (DVD) and Digital Video Broadcast (DVB) decoder, in accordance with one

20 embodiment of the present invention, includes a stream demultiplexer, a data storage buffer and a control Central Processing Unit (CPU). The stream demultiplexer demultiplexes and depacketizes the raw DVD data stream, subsequently stores the video and audio data in the

25 storage buffer and informs the CPU thereabout.

The stream demultiplexer extracts the timing information embedded in each data pack of a data stream and, accordingly, generates ~~a tag~~ containing the decode time stamp, the presentation time stamp and the storage

30 location of each data pack stored in the buffer. messages about the stored data and their locations in the storage buffer. These messages include tags

-2-

The CPU reads the information recorded in each tag and, accordingly, generates task definition packets for use by the decoder.

The decoder is fully synchronized with respect to
5   a signal generated by a video output processor of the system. Accordingly, in a steady state and under normal operating conditions, the CPU is interrupted only when the synchronization signal arrives thereby significantly reducing the number of interrupts that
10   the CPU must service. Furthermore, the CPU benefits from having an entire period of the synchronization signal to service the interrupts and therefore has relaxed timing requirements.

The decoder is pipelined to reduce the idle time
15   and hence to increase the utilization of the CPU and the video decoder. Accordingly, during each cycle of the synchronization signal, while the video decoder decodes the data, the CPU generates a task definition packet for the data to be decoded during the next
20   cycle.

Because the stream demultiplexer removes all the timing information from the data before storing it them in the storage buffer, the tasks performed by the audio and the video decoders are simplified. Furthermore,
.25   because the CPU is the component that makes all the decisions about the particular data and its their decode time, the audio and video decoders have vastly simplified tasks.

The stream demultiplexer also depacketizes the
30   audio data stream. The stream demultiplexer extracts

the timing information from each audio data pack and stores its payload in the buffer. Thereafter, the audio decoder determines the offset between the beginning of an audio data packet and the beginning of an audio

5   frame and supplies the information to the CPU. The CPU using the time stamp information provided by the stream demultiplexer and the offset information provided by the audio decoder determines the presentation time of an audio frame.

10   Other aspects and advantages of the present invention are apparent from the following detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

15   Fig. 1 shows a block diagram of a decoder in accordance with one embodiment of the present invention.

Fig. 2A shows the various components of a DVD data pack.

20   Fig. 2B shows the various components of a packet of data pack of Fig. 2A.

Fig. 2C shows the various components of a DVB transport data packet.

Fig. 3 shows the basic components of a task

25   definition packet in accordance with one embodiment of the present invention.

Fig. 4 shows the format of a message generated by the audio decoder to the CPU when a sync word in the audio stream is found.

30   Fig. 5 shows an audio buffer in accordance with

-4-

one embodiment of the present invention.

Fig. 6 shows the connections of the stream demultiplexer to various modules in the decoder of Fig. 1 in accordance with one embodiment of the present invention.

Fig. 7 shows an event tag format in accordance with one embodiment of the present invention.

Fig. 8 shows a DVD-oriented layered-packetized protocol that is transported by the stream demultiplexer of Fig. 1 in accordance with one embodiment of the present invention.

Fig. 9 shows a message queue in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

A schematic block diagram of a Digital Versatile Disc (DVD)/Digital Video Broadcast (DVB) decoder 20, in accordance with one embodiment of the present invention, is illustrated in Fig. 1.

Decoder 20 uses three data paths in either the DVD or DVB mode of operation, namely a video data path, an audio data path, and a control path.

Decoder 20 includes, in addition to other components not shown, a Network Port (NP) 22, a DVD-DSP interface 24, a Stream Demultiplexer (SD) 26, a timer 30, a clock generator 32, an audio decoder 34, a video decoder 36, an Audio Output Processor (AOP) 38, a Video Output Processor (VOP) 40, a control Central Processing Unit (CPU) 54 and a register 56.

In decoder 20, only a subset of the above

components process the audio data. For example, audio decoder 38 only decodes encoded audio data. Similarly, the processing of video data is performed by a subset of the decoder 20 components. The processing of video

5   data when decoder 20 is in the DVD mode of operation is described first.

The video data path of decoder 20 decodes and displays MPEG (MPEG1 or MPEG2) (developed by the Moving Pictures Expert Group of the International Standards

10   Organization (ISO)) compressed video data synchronously with timer 30.

As shown in Fig. 1, the video data path includes DVD-DSP interface 24, SD 26, timer 30, video decoder 36, VOP 40, CPU 54 and register 56. In particular,

15   DVD-DSP interface 24 retrieves raw DVD data bit streams and reformats it into bytes before supplying it to SD 26. SD 26 demultiplexes and depacketizes the video data stream and transports compressed video data to the compressed video buffer in buffer 48. Video decoder 36

20   fetches the compressed video data, decodes the data and stores the decoded data in one of the frame stores in buffer 50. VOP 40 retrieves the video frame data from buffer 50, merges the video frame data with OSD (on-screen display) and/or sub-picture data, and encodes

·25 the video frame data as an NTSC or PAL output for NTSC/PAL TV monitor 42.

In particular, depending on the data consumption rate, DVD-DSP Interface 24 retrieves raw data residing

on DVD-DSP 46 at a typical rate of approximately 11^ *megabits*

(Mb/s).

30   ~~Megabit~~ per second^ The DVD data stream supplied to SD

26 contains compressed audio, video, control and
synchronization information. SD 26 demultiplexes and
depacketizes the data stream, storing the demultiplexed
compressed audio and video data in data buffer 48, ~~which
is typically a First-In-First-Out (FIFO) buffer~~. SD 26
has access to 32 different addressable locations in buffer
~~FIFO~~ 48. A host programmable memory, called the routing
table (not shown in Fig. 1), directs SD 26 to a
particular location within buffer ~~FIFO~~ 48 in which the
elementary streams and substreams of data are stored.

Each DVD data is composed of data units, commonly
referred to as data packs. As illustrated in Fig. 2A,
each DVD data pack 200 contains a pack header 201 and
may contain one or more of the following elements: a
system header 202, an audio Packetized Elementary
Stream (PES) 204, a video PES packet 206 and a control
packet 208. As shown in Fig. 2B, each video PES packet
206 further contains a header 216 and a payload 214.
Each header 216 contains a PES control field 218 and
may contain a Decode Time Stamp (DTS) 210 or/and ~~or~~ a
Presentation Time Stamp (PTS) 212.

System header 202 identifies all the elementary
streams of a pack. SD 26 places the entire system
header 202 in ~~its own~~ buffer 48 for access by CPU 54.

SD 26 first demultiplexes the audio, video and
control components of the received DVD data.
Thereafter, SD 26 depacketizes the demultiplexed data,
namely, SD 26 removes PES control field 218, ~~header 208,~~ DTS 210 and PTS 212
from each packet 206 before storing the payload 214 in
buffer 48. Therefore, ~~audio decoder 34 and~~ video

decoder 36 ~~receive~~ *receives* only the payload 214 of data packet 206, which is an advantage of decoder 20. Because SD 26 demultiplexes the DVD data byte streams, several advantages are realized. First, CPU 54 is allowed to specialize in more complex and decision-based tasks. Second, unlike the control CPUs of conventional DVD decoders, CPU 54 is not required to have a high processing capability and is thus relatively inexpensive.

Fig. 2C shows the various components of a DVB transport data packet 230. Each DVB data packet 230 contains transport packet control 232, program identifier (PID) 234, adaptation field control 238, program clock reference (PCR) 236 and payload 240. Transport packet control 232 and PID 234 together form a transport packet header for the DVB data packet 230.

Concurrent references are made below to Figures 1-3 below. Each data pack 200 contains timing data, embedded in its pack header 201, which establishes a timing reference with respect to which the PTS 212 of the associated PES packet 206 is measured. In other words, the display time of a payload 214 is determined by comparing its PTS 212 to the reference timing data embedded in the associated system header 202.

Timer 30 is a real time clock that establishes the time base for all the timing references in system header 202. Therefore, timer 30 detects whether the recorded times in system header 202 have arrived. Timer 30 also includes facilities which trigger events when a particular time is reached. Consequently, timer 30

receives ● signal VsyncPhase from time ●30. VOP 40

generates events based on specific times or records the times when specific events occur.

As is seen from Fig. 1, Video Output Processor (VOP) 40 generates a *synchronization* signal, called Vsync, every 16

5   milliseconds *(i.e.,* 60 *Hz* for NTSC systems) which is the typical rate at which a Cathode Ray Tube (CRT) or a TV monitor is refreshed. Signal Vsync is supplied to Central Processing Unit (CPU) 54, video decoder 36 and timer 30. Signal Vsync synchronizes the activities of

10 the various components of decoder 20 and, accordingly, controls the DVD data consumption rate, as is described below.

Whenever SD 26 extracts the pack header 201 of a data pack 200, it instructs timer 30 to store the

15 current time in register 56. SD 26 stores the clock reference from the pack header in register 56. Therefore, register 56 stores both the clock reference of the data pack 200 as well as the time at which the pack arrives at SD 26. CPU 54 has access to register 56

20 and can read its content to determine the difference between the two timing data. In a steady state, when the system is fully synchronized, a fixed timing difference exists between the clock reference of data pack 200 and the time *when* data pack 200 arrives at SD

25 26.

Furthermore, after demultiplexing and depacketizing a DVD data pack 200, SD 26 extracts the associated time stamps of the data packet 206 (i.e. PTS 212 and DTS 210), stores *the* data *in* *video payload 214 buffer* 48 and

30 generates a *video message, or tag,* which contains the DTS and

the address of the stored data in ^buffer ~~FIFO~~ 48. In other
words, for each data pack 200, SD 26 generates a tag
containing information about both the decode time stamp
as well as the address of the stored data in ^buffer ~~FIFO~~ 48.

5   The generation of the tags is a significant advantage
of decoder 20.

The ^video messages, or tags, ~~tags~~ thus generated are made available to CPU
54 for further synchronization and control of decoder
20. Using the information stored in a tag, CPU 54

10   generates a Task Definition Packet (TDP) containing
instructions to video decoder 36 about the location of
the data in ^buffer ~~FIFO~~ 48. Upon the arrival of the next Vsync
signal, video decoder 36 decodes the data identified by
the TDP. If no TDP has been generated by CPU 54, video

15   decoder 36 does not decode any data at the occurrence
of the Vsync signal.

Fig. 3 shows the basic format of a TDP. Each TDP
is composed of 40 words (each word consists of 32 bits)
of which the last 5 words are reserved. The first word

20   of each TDP contains the address (with respect to
buffer 48) of the next video picture to be decoded.

Depending on the data consumption rate, CPU 54 may
instruct video decoder 36 to decode data out of
sequence. Therefore, if a faster data consumption rate

.25 is necessary, CPU 54 skips over the next frame of data
in ^buffer ~~FIFO~~ 48 and thus issues a TDP pointing to the
address of the succeeding data frame in ^buffer ~~FIFO~~ 48. If a
slower data consumption rate is necessary, CPU 54 does
not issue a TDP and thus the previously decoded data

30   frame is displayed.

Since the decision regarding when and which data to decode next is made by CPU 54 and not by video decoder 36, DVD/DVB decoder 20 provides several improvements over those of the prior art. First, video

5   decoder 36 benefits from a relatively simple design because it needs to decode data only when so instructed. Second, because CPU 54 is controlled through software, CPU 54 lends itself to a highly low-level control, thereby adding to the overall

10   flexibility of the system while making the system easily upgradeable. Third, the generation of the tags^ *relieves* ~~relives~~ CPU 54 from the computationally intensive and hence costly task of reading the data that flows between various components of decoder 20. The tags

15   enable the CPU to receive the information it needs to determine which data must be decoded, thereby greatly easing the computational burden on the CPU.

The tags generated by SD 26 are read by CPU 54 only when a Vsync signal arrives. Therefore, in a

20   steady state and under most operating conditions, CPU 54 is interrupted only when a tag is present at the occurrence of a subsequent Vsync signal; this is an advantage of the DVD/DVB decoder 20 which generates fewer interrupts than do other known systems. Moreover,

·25   because in the steady state, the interrupt requests are only generated during Vsync occurrences, CPU 54 advantageously has an entire Vsync time period^(i.e., the time interval between two successive Vsync. signals) to service those^ *interrupts.* ~~interrupt.~~ In particular, when a Vsync

30   signal arrives, CPU^ *54* reads^ *buffer* ~~FIFO~~ 48 to determine whether

SD 26 has generated any tags. If one or more tags exist, CPU 54 performs its assigned tasks and generates corresponding TDPs. However, CPU 54 has the entire period from the reading of the tags until the arrival

5   of the next Vsync signal to complete all of its tasks including the generation of the TDPs. Hence, CPU 54 may continue to finish its ongoing tasks uninterrupted, provided, however, that CPU 54 services the requested interrupt prior to the arrival of the next Vsync

10   signal. Therefore, decoder 20 benefits from very relaxed timing requirements. The relaxed timing requirements resulting from the availability of an entire Vsync period to CPU 54 to read the tags and to generate corresponding TDPs is a significant advantage

15   of DVD decoder 20 over the known DVD decoders which because they require their CPU control units ~~(e.g. CPU)~~ to immediately--but temporarily--abandon their ongoing tasks to service the interrupts in a very short time period, impose tight and rigid system timing

20   requirements.

       As stated above, although in a steady state and during normal operating conditions, the occurrence of the Vsync signal is the main synchronizing mechanism for generating interrupt requests to CPU 54, other

. 25   mechanisms such as polling may also be used to prompt the CPU to read the tags and generate TDPs.

       The data decoded by video decoder 36 is stored in buffer 50 for subsequent processing by VOP 40. Because CPU 54 decides when and which data is to be decoded

30   next, it also knows whether any decoded data has been

placed in buffer 50. Consequently, when such data

exists, at the occurrence of the next Vsync signal, CPU

54 instructs VOP 40 to fetch the data from the buffer 50

for further processing. After a known time period, the

5   data fetched and processed by VOP 40 is displayed on

monitor 42.

The generation and processing of the TDPs by CPU

54 and video decoder 36 are pipelined. In particular,

in a steady state, CPU 54 generates TDPs that will be

10   processed by video decoder 36 during the next Vsync

signal cycle. Therefore, CPU 54 is always generating

TDPs one Vsync signal cycle ahead. The pipelining

provides CPU 54 and video decoder 36 with an entire

Vsync signal cycle to respectively process the TDPs and

15   decode the data, while simultaneously reducing the idle

time of the decoder and thereby increasing the

utilization and throughput of the decoder.

The processing of audio data when decoder 20 is in

the DVD or DVB mode is described next. Referring to

20   Fig. 1, DVD-DSP interface 24 retrieves DVD data from

DVD-DSP 46, subsequently reformats this data as a

stream of bytes, and supplies the stream of bytes to SD

26. Similarly, Network Port 22 retrieves DVB data from

Network Interface 52, reformats this data as a stream

.25 of bytes, and supplies the stream of bytes to SD 26. SD

26 extracts the audio data stream from the incoming

data, depacketizes the data, and stores the data for

storage in buffer 48. Audio decoder 34 decodes the

stored audio data and writes the decoded data to data

30   buffer 50. Thereafter, AOP 38 fetches the data from

buffer 50, processes the data and supplies it to audio

Digital-to-Analog Converter (DAC) 44. Accordingly, in

this embodiment, the audio path includes DVD-DSP

interface 24, SD 26, timer 30, audio decoder 34, AOP 38

5   and CPU 54.

In MPEG, an audio time stamp is located at the

beginning of a PES packet, but it actually refers to

the first byte of the first audio sync frame that

begins in the packet, and thus, there is no alignment

10   between audio sync frames and PES packet boundaries.

~~Unlike the video bit streams, the~~ *The* audio bit streams do

~~not~~ contain ~~unique and identifiable~~ start code patterns

but they are not individually uniquely identifiable from the actual audio data, thereby

(sync words), ^making it difficult to detect the start

of an audio data frame.

15   In MPEG, a new audio frame is identified by

identifying an audio marker that constitutes a

repeating 12-bit field. Thus, for example, if the 12-

bit pattern of an audio marker is identified three

times, then it may be safely assumed, with a high

certainty,

20   degree of ^~~certainty~~ that the frame is an audio data

frame.

After depacketizing an audio frame, SD 26 stores

the PES payload in buffer 48 and generates ^~~a tag~~ *an audio message, or tag,*

informing CPU 54 of the location of the stored audio

48.

.25 frame in ~~the~~ buffer^ Consequently, SD 26 has knowledge

48

of both the location of the audio frame in ~~the~~ buffer^

as well as its corresponding time stamp. SD 26,

however, does not know the offset between the beginning

of the packet and that of the frame.

Audio decoder 34 decodes the data and upon detecting the corresponding sync word, marks the address and so informs the CPU 54. Figure 4 shows the format of the message generated by audio decoder 34 to

5   CPU 54 when audio decoder 34 finds a sync word in the compressed audio stream. Each such message has 32 ~bits,~ ~bytes~ Bits [31:20] of each word are reserved, and only the lower 20 bits are used.   The various parameters of the message when decoding MPEG audio are

10   defined below:

NCHANS	number of audio channels in the input bit-stream (including LFE)
SAMPRATE	sampling rate
LAYER	1=MPEG layer 1, 2=MPEG layer 2
INBUF_LEVEL	Level (count) of the compressed audio data input buffer
BITRATE	Bit rate in kilobits/sec
FRM_SIZE	Size of the last decoded frame
OUTBUF_WR_PTR	Current write pointer of the output buffer used to store decoded audio data
INBUF_RD_PTR	Current read pointer of the input buffer used to store compressed audio data
FRM_NUMBER	A running count of the frames decoded
STATUS	A status word used for debug and diagnostics
MSG_ID	Linearly incrementing number which acts as an ID for the message
MSG_TYPE	Has the value 3 indicating it is a Sync Found message.

Using

~~CPU 54 using~~ the marked address and the SD-
tags, CPU 54
generated ~~tags~~ establishes correlation between a sync
48.
word and its associated audio frame in ~~the~~ buffer.

5   Consequently, CPU 54 uses the information provided by

SD 26 and audio decoder 34 to determine when a

particular frame of audio data must be presented, the

details of which are described below.

Fig. 5 shows an audio buffer 100 in accordance

Audio buffer 100 is part of input buffer 48.
10   with one embodiment of the present invention. In audio

an SD-generated
buffer 100, pointer 102 shows where the address part of ^
106
~~SD 26~~ tag points. *Arrow 104 indicates a sync frame for that tag 106.*

The audio decoder 34 of Fig. 1 supplies the

location of the audio sync word in system memory (e.g.,
a
15   in ~~the~~ DRAM). When audio decoder 34 detects a sync

word, it captures the current system memory address

from which it is reading. Sync frames are generally of

a fixed size as indicated by audio buffer 100 of Fig.

5. Thus, if CPU 54 knows the start address, it can

20   obtain other start addresses by simply adding a

multiple of the sync frame size.

Further, CPU 54 knows the nominal compressed audio

data rate. Thus, once CPU 54 has a time stamp, it can

add a time offset per byte consumed by AOP 38 to obtain

·25 the desired audio decode time. By comparing the

desired decode time with the actual system time, CPU 54

determines whether audio decoder 34 is ahead of or

behind schedule. CPU 54 can then instruct audio decoder

34 to skip or repeat data as appropriate.

30   Referring to Fig. 1, timer 30 may perform timer

operations for the audio data processing that are similar to its timer operations for the video data processing, as described above.

In some embodiments, audio decoder 34 is

5 controlled by CPU 54. Under CPU 54 control, audio decoder 34 may operate as a slave co-processor or as a free-running decode engine. In slave co-processor mode, audio decoder 34 decodes one or more audio frames in response to a CPU 54 decode command. A new command is

10 required for every frame (or a number of frames) to be decoded. Upon completion of its tasks, ^audio ~~Audio~~ decoder 34 notifies CPU 54 of the completion of its activities by *(See Fig. 9 discussed below)* storing a message in the message queue ^of buffer 48. In some embodiments, audio decoder 34 generates an

15 interrupt request to CPU 54 after completing its tasks. Thereafter, audio decoder 34 remains idle until it receives another command.

In free-running mode, audio decoder 34 decodes audio frames when the amount of data in ~~the~~ input
48
20 buffer^reaches a predetermined threshold level. In this mode, CPU 54 may only generate commands to synchronize or adjust the data consumption rate. Thus, the audio decode rate is regulated by the levels of ~~the~~ input
48   *50*
buffer^and ~~the~~ output buffer^and the audio decode
·25 operation will be suspended while the input buffer content level is too low or the output buffer level is too high relative to predetermined threshold levels.

In addition to data processing, audio decoder 34 maintains audio synchronization by providing CPU 54

30 with the current positions of the input and output

buffer pointers at the beginning of each audio frame. Based on this information, CPU 54 correlates the encoded and actual presentation time of the audio data and provides audio synchronization adjustment commands

5   as appropriate.

Start-up processing typically occurs on a reset, power-on, or while switching programs. CPU 54 mutes the audio output until audio presentation time stamps arrive in the data stream, and the audio

10 synchronization software is running.

CPU 54 manages the total delay in the audio decode-presentation path by manipulating the depth of the audio ^input ^and output ^buffers. In particular, CPU

portions of   buffer 48   buffer 50.

54 may control the buffer depth by controlling when

15 audio decode tasks are issued and started. Audio decoder 34, in free-running mode, holds off from processing the next audio frame until the input buffer level reaches a predetermined threshold level (e.g., as provided by CPU 54). The audio decode-presentation path

20 delay remains constant and matches the video decode-presentation path delay.

In steady state mode, minor synchronization adjustments may be required at various times. Due to the nature of audio decoding, there are three different

·25 granularities of delay adjustment ^including, adding or

including:

dropping entire audio frames, adding or dropping multiples of thirty-two samples from a frame, and delay adjustments made by changing the audio DAC clock (not shown) for controlled periods of time, a "micro-

30 stepper" approach. Because the audio DAC clock tracks

the transport stream, adjustment of the audio decode
loop should be fairly infrequent.

The audio decoding operation also supports bit
stream error handling. In particular, the audio
5   decoder 34 detects the bit stream errors by means of a
CRC (Circular Redundancy Check) provided in the audio
stream. When an error is detected, error concealment
is applied to minimize the audible effect of the error.
AOP 38 reads the decoded audio data, further processes
10   the data and subsequently supplies the data to DAC 44.

In some embodiments, AOP 38 interfaces to
commercially available two-channel and six-channel
audio DACs. AOP 38 retrieves data from buffer 50 and
transmits the data to external audio DAC 44 at a rate
15   dictated by the DAC clock. Based on a write request
from audio decoder 34 and a read request from AOP 38,
the MMU (memory management unit) a memory unit (discussed below with
respect to Fig. 6) maintains a read pointer and a write
pointer for the audio output buffer in buffer 50. By
20   monitoring the buffer depth, rate of change, and
direction of change, CPU 54 determines whether the DAC
clock should be increased or decreased in frequency
relative to the video (i.e., pixel) clock (not shown).

Referring to Fig. 1, decoder 20 of Fig. 1 also
25   provides a digital video broadcast (DVB) operation. In
DVB mode (of operation), decoder 20 of Fig. 1 receives
a DVB data stream from the external demodulator,
selects the proper video and audio programs, extracts
timing and compressed video and audio data, decodes the
30   data, and presents the data via NTSC/PAL TV monitor 42

and audio DAC 44.   Similar to the DVD operation as
described above with respect to Fig. 1, decoder 20 in
DVB mode extracts system information from the DVB data
stream and places the system information in buffer 48
5   for use by CPU 54.

In particular, the DVB video path operation is
similar to the DVD video path described above with the
following exceptions.   In the DVB video path, DVB data
arrives at Network Port (NP) 22 from a network
10   interface 52 at 40 Mb/s ~~(megabits per second)~~ instead
of at DVD-DSP interface 24 from DVD-DSP 46 at 11 Mb/s.
Also, in the DVB video path operation, SD 26 may
process a transport stream instead of a program stream.
In addition, in the DVB video path operation, VOP 40
15   does not support sub-pictures.

In particular, NP 22 receives data from the
external demodulator network interface 52 and reformats
the data as a series of bytes. NP 22 then sends the
data to SD 26.

20   In the video path of the DVB mode, SD 26 operation
is generally similar to the DVD mode at the PES packet
level and elementary stream level.   However, at the
transport packet level, SD 26 operation differs from
the DVD mode of operation as described below.   SD 26
·25   looks for periodic repetitions of the sync field in
transport packets and achieves byte alignment by
synchronizing itself to the occurrences of packets in
the transport packet layer.   Also, SD 26 demultiplexes
transport packets based on the PID they contain.   SD 26
30   then depacketizes the transport packets into a stream

of PES packets, extracting control and navigational
information. In particular, for the video path, the
splice information contained in the transport
adaptation field is extracted. CPU 54 receives this

5   extracted information, through the messaging system as
described above, and uses the information to determine
when to force timer 30 to reset the local time to match
the input stream clock reference. In one embodiment,
the navigational information appears in the form of SI

10   table sections which SD 26 places into buffer 48 for
use by CPU 54, as discussed further below with respect
to Fig. 7.

In DVB mode, the audio path operation is similar
to the video path operation except that audio decoder

15   34 and AOP 38 replace video decoder 36 and VOP 40,
respectively. Also, the operation of audio decoder 34
and AOP 38 is generally similar in the DVD mode and the
DVB mode.

Finally, in the DVB mode, the DVB clock control

20   path operation is similar to the DVD clock control path
operation described above with the following exceptions
described below. First, data comes through NP 22
instead of DVD-DSP interface 24. Second, SD 26
processes a transport stream rather than a program

·25   stream. SD 26 extracts a clock reference from the
incoming^transport stream as similarly described above for the
DVD operation. However, in DVB mode, there can be
several clock references. Thus, SD 26 only extracts
program clock references (PCR) from transport packets

30   that have a certain PID. In particular, even though SD

26 may demultiplex and depacketize other PID packets with PCRs, only the programmed PID will be used for clock reference operations. Thus, the actual operations with clock *references* are similar to *that* described above in the DVD mode. Unlike the DVD mode in which the data stream is pulled into the system, *in the DVB mode* the clock frequency is adjusted to match the data rate of the incoming data stream.

5

In one embodiment, a DVD/DVB decoder system that includes decoder 20 of the present invention may be implemented using an ASIC (application specific integrated circuit), an on-chip processor (for CPU 54), 2 or 4 megabytes of conventional 16 megabit (MB) SDRAM arranged as 2x512Kx16, expandable to 16MB, a ROM (0.5 to 16 MB) for *the* operating system, application, driver, and diagnostic instruction and data storage, a stereo audio DAC or *a* 6 channel DAC *and/or an* S/PDIF output for AC-3 playback, and an 8052 microcontroller for front panel, infrared (IR) and general purpose input/output (I/O).

10

15

20

Fig. 6 shows SD 26 connections to other modules in decoder 20 of Fig. 1 in accordance with one embodiment of the present invention. In particular, SD 26 connects to timer 30, a memory management unit (MMU) *60*, a host bus interface *62*, and a network port/DVD controller *64*.

.25

Referring to Fig. 6, as part of depacketizing and transporting the byte stream, SD 26 writes the extracted information to MMU *60*. MMU *60* subsequently transfers the information into the system memory (e.g.,

30

buffer 48 of Fig. 1). Also, the audio decoder 34 of

Fig. 1 and the video decoder 36 of Fig. 1 request audio

and video data, respectively, from the system memory

(e.g., ~~the~~ buffer 48 of Fig. 1) using MMU ~~50~~ 60. In one

5   embodiment, the system memory includes thirty-two

separate buffers (e.g., buffer 48 includes thirty-two

separate sub-buffers): one buffer each for MPEG video,

audio, sub-picture, teletext, and various other data

and control streams. Further, SD 26 tags certain events

10   in the bit stream being written to system memory with,

for example, the desired decode or presentation time,

the current write location in system memory, etc. SD

26 presents the tags to CPU 54 through the message

queue (e.g., an event FIFO) stored in buffer 48 of

15   Fig. 1.

For example, when SD 26 detects clock reference (CR)

fields ~~(CR)~~ in the input stream, SD 26 provides the

value to timer 30. Timer 30 may use this data to set

the current system time when decoder 20 of Fig. 1 is

20   attempting to gain initial synchronization. When the

system (decoder 20) is already synchronized, SD 26 can

capture the current time from the timer 30 when SD 26

detects CR fields and store the CR fields in registers

for presentation to CPU 54.

.25   SD 26 extracts system time stamps, either PCR for

transport streams or SCR for program streams, and ~~SD~~

puts these extracted time stamps in host-accessible

registers along with the current system time and the

level of each^ buffer ~~FIFO~~ in the^ received ~~receive~~ packet. Using the

30   time stamp and^ buffer ~~FIFO~~ level versus system time data as

phase measurements into a software-controlled phase-locked loop, CPU 54 can adjust the frequency of the video pixel clock and the audio oversampling clock so that the system time base matches the source material time base on average. The adjustment will lower the rate of skip/repeat events in the presentation processes. When splices occur in a transport stream, there may be a discontinuity between time stamps on either side of the splice.

For transport streams, SD 26 separates a particular program from the remaining programs in the stream. Generally, a program will consist of at most one video elementary stream, at most one audio elementary stream, possibly a teletext stream or a sub-picture stream, and a number of streams containing system information tables. A stream as defined by a service provider may have more than one elementary video or audio stream, but decoder 20 of Fig. 1 only decodes one of each type of stream at a given time.

SD 26 accepts PES packets, SI or PSI tables, or program stream system headers from NP 22. SD 26 places depacketized data into one of a number of circular buffer tags supplying pointers to units within a buffer. For example, for video data, SD 26 provides pointers to picture start codes. For audio data, there are pointers to packets with time stamps present in the stream. For program streams, there generally will be a single default PID used for routing purposes.

Accordingly, SD 26 of Fig. 6 selectively transports demultiplexed and depacketized byte streams.

For example, if decoder 20 only includes one audio
decoder 34 as shown in Fig. 1, then SD 26 only
transports audio data for the user programmed language
for audio output (e.g., English or *Japanese assuming*
~~Japanese) (assuming~~
5   that the audio decoder can only decode audio data for
one language at a time).

Fig. 7 shows an event tag format in accordance
with one embodiment of the present invention. SD 26
generates tags that are stored in buffer 48 of Fig. 1.
10   In another embodiment, SD 26 generates an interrupt
after it writes a new tag to MMU *60* ~~50~~ of Fig. 6. In
particular, the tags contain addresses in the system
memory (e.g., in a RAM, DRAM, or SRAM that includes
buffer 48 of Fig. 1) where data associated with certain
15   events in the input stream can be found. Thus, Fig. 7
provides a table for an event tag that lists the event
tag bit definitions in accordance with one embodiment
of the present invention.

Fig. 8 shows a layered-packetized protocol that is
20   demultiplexed, depacketized, and transported by SD 26
of Fig. 1 in accordance with one embodiment of the
present invention. In particular, a byte stream *80* ~~100~~
represents a layered-packetized protocol that includes
PCI packets, DCI packets, video packets, audio packets,
.25 and DSI packets. For example, MPEG represents a
layered-packetized protocol. In one embodiment, SD 26
of Fig. 1 can handle the transport layer, the PES
layer, and the table layer of a layered-packetized
protocol such as MPEG2, and CPU 54 can handle parsed
30   elementary stream layers and audio/video streams. A

consisting of the above-⬤ntioned audio and video messages generated by SD 26 for use by CPU 54.

layered-packetized protocol typically includes header information and payload information in the packets of data, and the header information contains various information respecting the packet or perhaps subsequent

5   or preceding packets of similar or different packet types.   In particular, SD 26 can use the header information to selectively demultiplex packets of byte stream 80, 100 as discussed above with respect to Figs. 1 and 2A - 2C.

10   Fig. 9 provides a message queue 120 106 in accordance with one embodiment of the present invention.   Message queue 120 106 includes tags.   Message queue 120 106 is a FIFO queue stored in a sub-buffer of buffer 48 of Fig. 1. In particular, when SD 26, as discussed above with

15   respect to Figs. 1 and 2A - 2C, identifies significant information such as the location of the start of a new video frame or a presentation time stamp of a particular video frame, then SD 26 generates a tag (video message) such as a tag 122 108 and stores tag 122 108 in message queue 120 106.

20   In one embodiment, SD 26 generates a message that audio or video data is stored in buffer 48 of Fig. 1 and ready for decoding even prior to storing all the audio or video data, respectively, in buffer 48.   CPU 54 periodically accesses message queue 120 106 (e.g., every

.25   Vsync) and reads the tags in real time. In particular, message queue 120 106 provides a FIFO queue such that the most recent event is stored in the tag that is at the top of the FIFO queue.   For example, referring to Fig. 8, if video data is identified prior

30   to audio data in byte stream 80, 100, then a tag (video message) in message

queue ^106 *120* identifying a time stamp for the video data

of byte stream ^100 *80* would be lower in message queue ^106 *120*

than a tag ^identifying *(audio message)* a time stamp for the audio data

(e.g., such tags may also include an address

5   identifying the location of the audio and video data,

respectively, in buffer 48 of Fig. 1).

    Accordingly, SD 26 performs time/space

multiplexing that allows CPU 54 to function in a batch

mode rather than generating interrupts to CPU 54 each

10   time a significant piece of information is identified

by SD 26. For example, message queue ^106 *120* as used by SD

26 represents an alternative approach to having SD 26

generate interrupts to CPU 54 each time SD 26

identifies significant information (the interrupt

15   approach). The interrupt approach is inefficient,

because every time an interrupt is generated for CPU

54, CPU 54 spends a few cycles processing the

interrupt. Thus, SD 26 advantageously uses message

queue ^106 *120* to implement a hardware-based messaging

20   system (hardware-based tagging mechanism).

    Moreover, because CPU 54 is processing information

one Vsync signal cycle ahead, CPU 54 typically can

access the tags in message queue ^106 *120* and process the

tags appropriately to program audio decoder 34 of Fig.

·25   1 or video decoder 36 of Fig. 1 appropriately in real

time. For example, the MPEG protocol assumes that

there is at least one frame worth of delay such that

the presentation time stamps allow for such delay upon

receipt of the video data and the audio data.

30   Accordingly, decoder 20 uses SD 26 and message queue ^ *120*

-27-

to take advantage of this scheme by providing a

hardware-based messaging system for decoder 20 of

Fig. 1.   In one embodiment, CPU 54 uses MMU 60 of Fig.

6 to request all the messages stored in message queue 120₉

5   106 and CPU 54 performs this task periodically (e.g.,

every Vsync) such that the tags are processed well in

advance as necessary for appropriate display timing.

Alternatively, SD 26 may in all or some cases also

raise an interrupt to CPU 54 when generating a tag for

10   message queue 120.

Accordingly, SD 26 of Fig. 1 demultiplexes,

depacketizes, and transports byte streams and provides

a hardware-based tagging mechanism in accordance with

one embodiment of the present invention.   CPU 54 reads

15   the tags and, based on the tags, programs decoder 20 of

Fig. 1 such that the audio data and video data are

output within the presentation time requirements.

Although particular embodiments of the present

invention have been shown and described, it will be

20   obvious to those skilled in the art that changes and

modifications may be made without departing from the

present invention in its broader aspects. The appended

therefore the appending claims are to encompass within

their scope all such changes and modifications that

25   fall within the true scope of the present invention.